OPTIMASI SISTEM INFORMASI PENJADWALAN KULIAH BERBASIS *HEURISTIC SEARCH* YANG DIKOMBINASIKAN DENGAN TEKNIK *SMART BACK TRACKING* DAN *LOOK AHEAD* (STUDI KASUS PADA STMI – KEMENTERIAN PERINDUSTRIAN)

Dedy Trisanto[(1)], Muhamad Agus[(1)]

[(1)]Program Studi Sistem Informasi – STMI, Kementerian Perindustrian, Jakarta

ABSTRACT

Scheduling lecture is scheduled number of components consisting of courses, lecturer, students, classrooms, and time with a number of restrictions and requirements (constraints) certain to get optimal results and the best. In this paper will be discussed and created scheduling lecture with a problem-solving approach to the science of Artificial Intelligence (Artificial Intelligence), by using an approximation of the mathematical problem that is aiming to find a situation or object that meets a number of requirements or specific criteria (Constraint Satisfaction Problem) to get the optimal scheduling and the best. To solve these problems the solution search techniques used by an algorithm that will result in optimal scheduling and the best (heuristic search) techniques combined with Smart Backtracking and Look Ahead called Intelligent Search to find and resolve problems when encountered a condition where no there is a solution in due course scheduling constraints and requirements are not met (deadlock). The application of these methods and techniques in the course scheduling information system is built, using the PHP programming language and MySQL database to solve the problem of scheduling to get optimal results and the best.

Keywords: scheduling lecture, heuristic search

1. PENDAHULUAN

Penjadwalan kuliah merupakan pekerjaan yang tidak mudah. Terdapat berbagai aspek yang berkaitan dalam penjadwalan tersebut yang harus dilibatkan dalam pertimbangan di antaranya:

a. Terdapat jadwal-jadwal di mana dosen yang mengampu mata kuliah tidak bisa mengajar baik karena sedang tugas belajar di jenjang lebih tinggi maupun karena sudah memiliki jadwal mengajar di program studi lain.
b. Terdapat jadwal-jadwal yang telah ditentukan oleh pihak laboratorium untuk kelas-kelas tertentu.
c. Tidak boleh ada jadwal mata kuliah yang bersamaan atau bersinggungan dengan jadwal kuliah angkatan sebelumnya maupun sesudahnya
d. Distribusi jadwal mata kuliah diharapkan dapat merata tiap harinya untuk setiap kelas.
e. Banyaknya kelas per angkatan dan perprogram studi.

Maka pada implementasi penjadwalan mata kuliah diperlukan suatu teknik yang baik untuk permasalahan penjadwalan tersebut. Jadi dengan komputasi pada komputer yang cepat saja tidak cukup. Dengan pemanfaatan teknik yang bagus maka akan didapat hasil yang optimal dan terbaik.

2. TEORI DASAR

Constraint Satisfaction Problem (CSP) merupakan sebuah pendekatan dari problem yang bersifat matematis dengan tujuan menemukan keadaan atau obyek yang memenuhi sejumlah persyaratan atau criteria. Sebuah constraint diartikan sebagai sebuah batasan dari solusi memungkinkan dalam sebuah problem optimasi. Algoritma yang menjadi kandidat dalam mencari sebuah solusi CSP, antara lain:

a. Backtracking
b. Forward checking
c. Look Ahead
d. Constraint propagation
e. Arc and path consistency
f. Variable and value ordering
g. Hill climbing

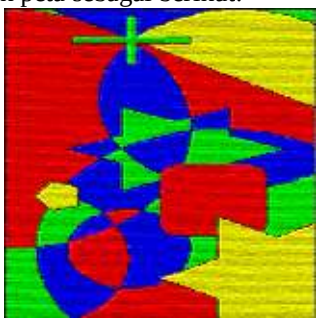
Dengan algoritma diatas dapat menganalisis banyak hal, diantaranya:

a. Kompleksitas waktu
b. Kompleksitas ruang
c. Terminasi / Kelengkapan
d. Optimasi

Banyak problem dapat dikategorikan sebagai CSP, diantarany mewarnai peta (*map coloring*).

Diberikan sebuah peta planar (dalam satu bidang), dan diberikan asumsi bahwa kita hanya dapat mewarnai dengan sejumlah k warna, warnailah peta

sehingga tidak ada 2 bidang bertetangga yang memiliki warna sama. Contoh pewarnaan dengan 4 warna untuk peta sebagai berikut:



Gambar 1 Pewarnaan Peta (*Map Coloring*)

Dua bidang disebutkan sebagai bertetangga jika ada bagian dari batas wilayah yang bersinggungan, dan saling menyambung. Untuk peta di atas, jelas bahwa tiga warna tidak akan cukup, karena ada satu wilayah yang dikelilingi dengan lebih dari 3 wilayah lainnya. Problem di atas dapat ditangani oleh program computer, namun demikian pembuktian secara matematisnya tidak dapat dilakukan secara langsung.

2.1 Representasi CSP

CSP biasanya direpresentasikan dengan sebuah graf, tanpa arah, disebut sebagai *Contraint Graph*, dengan node-nya adalah variable dan jalurnya adalah batasan yang dimiliki oleh node. Untuk batasan tunggal, dapat dilengkapi dengan mendefinisikan ulang domain yang ada sehingga mengisi variabel tersebut. Constraint dengan batasan yang lebih tinggi dapat dinyatakan dalam arc (jalur berarah).

Sebuah constraint akan dapat mempengaruhi satu atau lebih variabel (1..n) dalam definisi permasalahan. Jika semua *constraint* dalam CSP adalah biner (ada minimal 2 variabel kemungkinan untuk solusi berikutnya), maka semua variabel dan constraint dapat direpresentasikan dalam sebuah graf, dan algoritma CSP dapat diberlakukan untuk mengeksploitasi graf.

Konversi dari sebuah problem CSP ke dalam graf binernya, dilandasi ide untuk memperkenalkan sebuah variabel baru yang mengenkapsulasi himpunan variabel yang memiliki *constraint*. Variabel baru ini merupakan hasil "perkalian" *Cartesian* antara *domain* dengan setiap variabel. Perkalian ini akan membentuk kombinasi antara domain dan variabel, yang merupakan himpunan terbatas (meskipun bisa sangat banyak kombinasinya).

Sekarang, setiap n-ary *constraint* dapat dikonversi ke dalam *constraint* tunggalnya, yang mengenkapsulasi variabel awalnya. Dari *constraint* tunggal ini, sebuah batasan terhadap variabel akan dapat dicari solusinya. Dengan demikian setiap n-ary *constraint* dapat disubtitusikan dengan variabel yang sesuai di dalam domainnya. Hal ini tentu menarik sebab semua CSP tentu akan dapat direpresentasikan ke dalam binary *constraint*-nya.

contoh tentang *crossword.* Kita memiliki variabel sebagai berikut:

Tabel 1 Variable Crossword

VARIABLE	STARTING CELL	DOMAIN
1ACROSS	1	{HOSES, LASER, SAILS, SHEET, STEER}
4ACROSS	4	{HEEL, HIKE, KEEL, KNOT, LINE}
7ACROSS	7	{AFT, ALE, EEL, LEE, TIE}
8ACROSS	8	{HOSES, LASER, SAILS, SHEET, STEER}
2DOWN	2	{HOSES, LASER, SAILS, SHEET, STEER}
3DOWN	3	{HOSES, LASER, SAILS, SHEET, STEER}
5DOWN	5	{HEEL, HIKE, KEEL, KNOT, LINE}
6DOWN	6	{AFT, ALE, EEL, LEE, TIE}

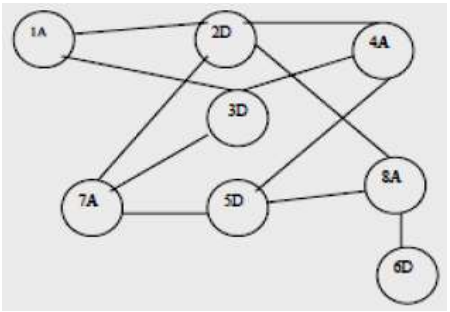
Domain dari setiap variabel adalah daftar kata yang "kemungkinan" merupakan isi dari variabel tersebut. Sehingga diketahui bahwa untuk variabel 1ACROSS, memerlukan 5 huruf, 2DOWN lima huruf, 5DOWN memerlukan 4 huruf, dst. Perhatikan bahwa di dalam setiap domain terdapat 5 kemungkinan solusi, dan ada 8 variabel, dengan demikian jumlah keadaan yang memungkinkan adalah 5^8 = 390.625.

Semua *constraint* dapat direduksi ke dalam bentuk biner:

```
      1ACROSS[3] = 2DOWN[1]  i.e. the third letter of 1ACROSS must be
equal to the first letter of 2DOWN
      1ACROSS[5] = 3DOWN[1]
      4ACROSS[2] = 2DOWN[3]
      4ACROSS[3] = 5DOWN[1]
      4ACROSS[4] = 3DOWN[3]
      7ACROSS[1] = 2DOWN[4]
      7ACROSS[2] = 5DOWN[2]
      7ACROSS[3] = 3DOWN[4]
      8ACROSS[1] = 6DOWN[2]
      8ACROSS[3] = 2DOWN[5]
      8ACROSS[4] = 5DOWN[3]
      8ACROSS[5] = 3DOWN[5]
```

Gambar 2 Reduksi Bentuk Biner (*Crossword*)

Dari hasil reduksi permasalahan, dapat dibentuk gra f constraint, sebagai berikut:



Gambar 3 Graf Constraint (*Crossword*)

Pencarian Solusi CSP

Dalam bagian berikut akan dibahas lima metode umum dalam pencarian solusi CSP, yaitu: *geneate*

and test, backtracking, consistency driven, forward checking dan *look ahead.*

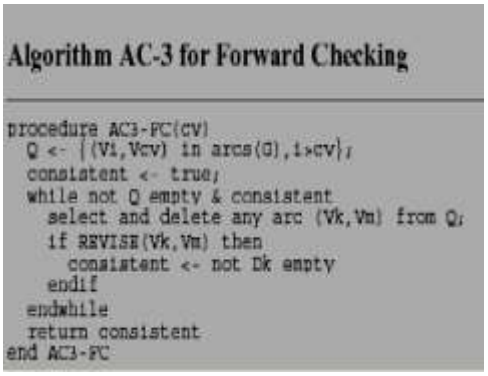
Generate and Test
Melalui cara ini, kita harus membangkitkan satu persatu alokasi variabel sehingga memenuhi semua *constraint*-nya. Struktur program untuk cara ini sangat sederhana, hanya berupa konstruksi loop, satu untuk setiap variabel, dan setiap constraint. Metode ini tidak efektif karena memiliki kompleksitas waktu yang sangat besar.

Backtracking
Dalam metode ini, diperlukan penyusunan ulang dalam urutan pengisian variabel. Cara yang paling efektif adalah dengan mencari solusi untuk variabel dengan constraint terbanyak, atau dengan domain yang paling sedikit. Urutan akan sangat menentukan cepat atau lambatnya solusi ditemukan. Algoritma dimulai dengan mengisikan variabel dalam *constraint*-nya, kemudian melakukan evaluasi terhadap *constraint*, apakah terpenuhi atau tidak. Lakukan hal yang sama, sampai semua variabel terisi. Jika variabel tidak dapat diisikan, maka harus dilakukan penelaahan ulang (*backtracking*), ke node diatasnya, atau variabel sebelumnya.

Consistency Driven Techniques
Teknik konsistensi secara efektif menyingkirkan banyak alokasi variabel yang inkonsisten diawal pencarian, sehingga akan mengurangi ruang pencarian. Teknik ini telah diujicobakan dan terbukti efektif untuk problem-problem yang bersifat hard (*NP-complete*). Teknik ini *deterministic*, yang berlawanan dengan sifat pencarian CSP yang *non-deterministic*. Dengan demikian, perhitungan yang *deterministic* dilakukan secepat mungkin, dan hanya melakukan perhitungan *non-deterministik* pada saat tidak ada lagi *propagasi* yang dapat dilakukan. Namun demikian, teknik konsistensi jarang diterapkan secara mandiri dalam pemecahan CSP.
Dalam CSP biner, berbagai teknik konsistensi untuk *graf constraint* akan diperkenalkan, yang dapat memotong ruang pencarian. Algoritma konsistensi akan dapat digunakan untuk mencari solusi parsial, dalam sebuah sub-pohon pencarian, yang dapat diperluas ke sub-pencarian berikutnya. Dengan cara seperti ini, jika ada potensi ketidakcocokan akan dapat dideteksi sedini mungkin.

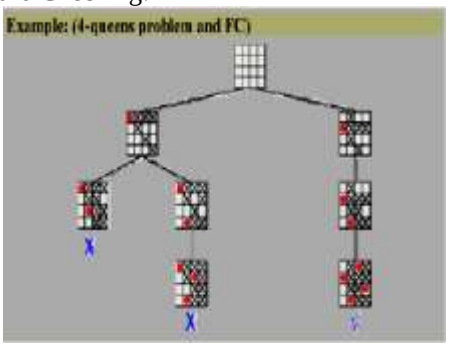
2.2 Forward Checking

Forward checking adalah cara termudah untuk menghindari konflik isi variabel. Sebagai pengganti konsistensi arc, untuk menginisialisasi nilai variabel, *forward checking* akan membatasi konsistensi *arc* ke dalam variabel yang belum diinisialisasi. Kita akan membicarakan tentang konsistensi *arc* yang dibatasi, karena melalui *forward checking* hanya akan dievaluasi *constraint* antara variabel saat ini dan variabel di depannya (berikutnya). Jika sebuah nilai dialokasikan untuk variable saat ini, maka nilai apapun dari variabel berikutnya, yang dapat membawa konflik pada variabel saat ini, akan disingkirkan (temporer) dari domain. Keuntungan dari hal ini adalah, jika *domain* dari variabel berikutnya kosong, maka akan segera diketahui bahwa solusi yang terbentuk sampai saat ini adalah inkonsisten. Dengan *forward checking*, maka percabangan dari pohon pencarian yang akan membawa kegagalan akan dipangkas lebih awal, dibandingkan dengan *backtracking*. Perhatikan gambar 4 dibawah ini, bahwa jika sebuah variabel baru dibuka, maka semua nilai dipastikan konsisten untuk variabel sebelumnya, dan dengan demikian evaluasi terhadap alokasi nilai yang telah dilakukan tidak perlu lagi.



Gambar 4 Algoritma Forward Checking

Forward checking akan mendeteksi ketidak konsistenan lebih awal dari *backtracking*, dan dengan demikian akan memangkas semua percabangan dalam pohon yang tidak konsisten. Ini akan mempersingkat waktu pencarian keseluruhan, namun harus dicatat bahwa *forward checking* memerlukan lebih banyak waktu pencarian, ketika setiap alokasi ditambahkan untuk solusi parsial saat ini. Berikut adalah contoh 4-queens problem and Forward Checking:



Gambar 5 Contoh 4-queens problem and Forward Checking

Pilihan untuk melakukan *forward checking*, biasanya akan selalu lebih baik ketimbang melakukan *backtracking.*

2.3 Look Ahead

Forward checking hanya mengevaluasi isi *constraint* saat ini dan variabel yang setelahnya, bagaimana jika dilakukan evaluasi konsistensi *arc* yang lengkap, yang akan memangkas nilai domain, dan menghilangkan konflik? Cara ini disebut dengan (*full*) *look ahead* atau *maintaining arc consistency* (MAC).

Keuntungan dari *look ahead* adalah bahwa dapat dideteksi konflik antara variabel berikutnya dan sebelumnya, sehingga mengijinkan dipangkasnya cabang pohon pencarian yang tidak berpotensi lebih awal dibandingkan forward checking. Sama seperti halnya forward checking, jika sebuah variabel dibuka, maka semua nilai yang telah dibuka dijamin konsisten dengan variabel sebelumnya, sehingga evaluasi terhadap alokasi variable sebelumnya tidak diperlukan lagi.

Perlu diperhatikan pula bahwa dengan look ahead, perlu dilakukan lebih banyak pekerjaan ketika sebuah alokasi variabel ditambahkan pada solusi saat ini dibandingkan dengan forward checking. Berikut adalah gambar dari algoritma look ahead:

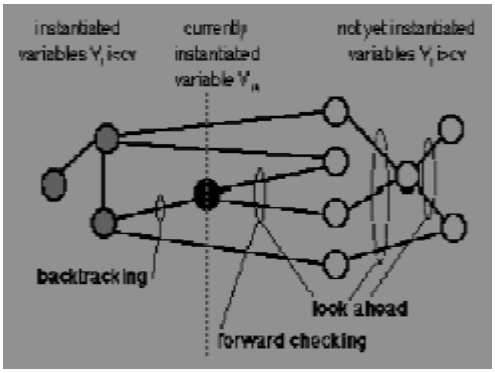
Algorithm AC-3 for Look Ahead

```
procedure AC3-LA(cv)
  Q <- {(Vi,Vcv) in arcs(G),i>cv};
  consistent <- true;
  while not Q empty & consistent
    select and delete any arc (Vk,Vm) from Q;
    if REVISE(Vk,Vm) then
      Q <- Q union {(Vi,Vk) such that (Vi,Vk) in
arcs(G),i#k,i#m,i>cv}
      consistent <- not Dk empty
    endif
  endwhile
  return consistent
end AC3-LA
```

Gambar 6 Algoritma Look Ahead

Perbandingan Teknik Propagasi

Gambar dibawah ini memberikan perbandingan evaluasi constraints yang terjadi dengan teknik-teknik yang telah dijelaskan sebelumnya.



Gambar 7 Graf Teknik Propagasi

Lebih banyak propagasi pada setiap node, akan menghasilkan pohon pencarian dengan lebih sedikit node, tapi memiliki biaya (kerja) yang lebih banyak, yang diperoleh dari biaya pemrosesan setiap node yang lebih mahal.

Dalam satu hal, memang terbukti bahwa membentuk jalur dengan n-konsistensi dari problem awal akan mereduksi kebutuhan untuk melakukan pencarian, namun akan lebih memerlukan biaya tinggi disbanding backtracking. Dari sinilah mengapa biasanya forward checking dan backtracking lebih banyak digunakan dibandingkan dengan look ahead.

Penyusunan Urutan Variabel dan Nilai

Sebuah algoritma pencarian untuk CSP memerlukan urutan pengisian variabel, demikian pula dengan mekanisme backtrackingnya, harus sesuai dengan urutan yang ditentukan sebelumnya. Memilih urutan yang tepat untuk variabel dan nilai dapat meningkatkan efisiensi solusi CSP.

2.4 Heuristik dalam CSP

Dalam beberapa tahun terakhir, pencarian local dalam bentuk *greedy search* (mengambil nilai heuristic terbaik), menjadi popular kembali. Dengan algoritma yang bersifat *greedy search*, nilai ketidak konsistenan akan digantikan seiring jalannya pencarian. Dengan metafora "perbaikan" perlahan (*hill climbing*), algoritma *greedy search*, akan mencapai solusi lengkap. Untuk menghindari adanya "lokal optima", algoritma harus dilengkapi dengan *heuristik* yang "mengacak" pencarian. Sifat *stokastik* ini secara umum akan mengurangi kelengkapan dari pencarian yang dihasilkan dalam metode pencarian sistematik.

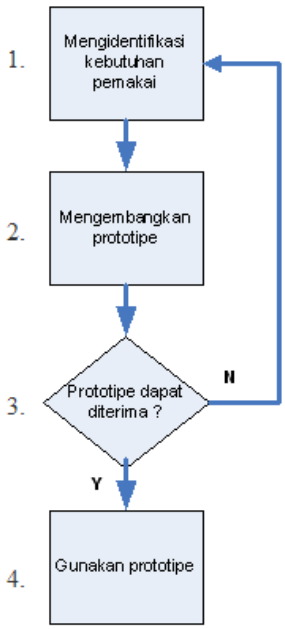
Metodologi pencarian lokal untuk CSP, menggunakan beberapa istilah berikut ini:

a. ***State* (*configuration*)**: sebuah alokasi yang memungkinkan untuk membentuk solusi untuk semua variabel, jumlah state adalah sama dengan hasil perkalian dari ukuran domain dan variabel.

b. ***Evaluation value***: jumlah constraint yang tidak memenuhi syarat (biasanya diberikan bobot).
c. ***Neighbor***: keadaan yang didapatkan dari keadaan saat ini, dengan menggantikan isi sebuah variabelnya.
d. ***Local-minimum***: sebuah state yang bukan merupakan solusi, dan memiliki nilai evaluasi tetangga yang semuanya lebih besar (secara *heuristik*) atau sama dengan nilai evaluasi state sekarang.
e. ***Strict local-minimum***: sebuah state yang bukan merupakan solusi, dan memiliki nilai evaluasi tetangga yang semuanya lebih besar (secara heuristik dibanding nilai evaluasi state sekarang.
f. ***non-strict local-minimum***: state yang merupakan *local-minimum* namun tidak mutlak

3. METODE PENELITIAN

Penelitian dilakukan dengan menggunakan metodologi pengembangan sistem yang dipilih adalah *Prototyping Evolusioner* dengan tahapan sebagai berikut:



Gambar 8 *Prototyping Evolusioner*

Penjelasan langkah – langkah pada gambar 8 secara deskriptif dijelaskan sebagai berikut:

a. Mengidentifikasikan kebutuhan pengguna yaitu dengan melakukan observasi terhadap sistem penjadwalan kuliah yang sedang berjalan dan mewawancarai pengguna sistem untuk mengetahui apa yang diminta dan dibutuhkan oleh pengguna sistem.
b. Mengembangkan atau membuat *prototype* sistem usulan. Untuk membuat *prototype* tersebut, maka digunakan bahasa pemrograman *PHP* dengan penerapan algoritma berbasis *heuristic search* yang dikombinasikan dengan teknik *smart back tracking* dan *look ahead* serta menggunakan *database MySQL* sebagai basis datanya.
c. Menentukan apakah *prototype* dapat diterima. Untuk mengetahui hal tersebut, maka dilakukan demonstrasi *prototype* aplikasi usulan kepada pengguna, apakah telah sesuai dengan kebutuhan pengguna. Jika iya, akan dilakukan langkah selanjutnya, dan jika tidak, *prototype* akan direvisi dan kembali ketahapan mengidentifikasikan kebutuhan pengguna untuk mendapatkan data-data yang diperlukan dalam perbaikan sistem usulan tersebut.
d. Setelah *prototype* diterima, maka tahap selanjutnya akan diserahkan ke pengguna, apakah *prototype* akan diterapkan atau tidak untuk perbaikan sistem lama.

4. HASIL DAN PEMBAHASAN

Ada tujuh tahapan / persiapan yang harus dilakukan dalam penerapan sistem informasi penjadwalan kuliah yang disusun secara otomatis oleh program, hal yang perlu diperhatikan dan dilakukan sebelum melakukan proses pembuatan jadwal kuliah secara otomatis adalah:

a. Penginputan data dosen jika belum terdaftar sebagai dosen pengampu mata kuliah, dengan tampilan sebagai berikut:

b. Penginputan mata kuliah yang akan diselenggarakan berdasarkan kurikulum permasing-masing program studi, dengan tampian sebagai berikut:

c. Pengelompokan mata kuliah berdasarkan angkatan dan semester yang akan diselenggarakan (paket mata kuliah), dengan tampilan sebagai berikut:

d. Penginputan nama kelas berdasarkan program studi dan waktu penyelenggaran kuliahnya (pagi atau malam), dengan tampilan sebagai berikut:

e. Penginputan data ruangan yang akan digunakan untuk perkuliahan dan maksimal daya tampung ruangan tersebut

f. Penginputan Shift (waktu kuliah)

g. Penginputan formulir elektronik kesanggupan dosen mengajar berdasarkan kesanggupan waktu dan mata kuliah yang akan dipilihnya.

Setelah tujuh tahapan / persiapan tersebut selesai, tahapan berikutnya adalah sebagai berikut:

a. Untuk proses penyusunan jadwal secara otomatis, user admin harus memasukan password terlebih dahulu untuk konfirmasi dan kemudian klik tombol "Mulai Susun Jadual>>". Dengan tampilan sebagai berikut:

Ketika User Admin mengklik tombol mulai penyusunan jadwal kuliah otomatis maka program akan mengecek terlebih dahulu kesiapan proses penyusunan jadwal apakah telah terpenuhi dan kemudian program akan mengecek batasan-batasan yang telah terprogram serta menampilkan error message jika terjadi kegagalan sistem dalam proses penjadwalan kuliah. Adapun listing yang berkaitan dengan hal tersebut terdapat pada page

mtk_kelas_verify.php, yang digunakan sebagai pemberitahuan ke user dibuat dalam bentuk array adalah sebagai berikut :

```
$err_msg_alokasi=array(1=>"Dosen belum diisi",
                       "Kelas belum diisi",
"Waktu kuliah salah (terlalu pagi atau terlalu
                       malam)", "Ruangan belum diisi",
"Dosen ini melebihi kuota SKS per
dosen.<br>\nSKS yang telah didapat = $sks, kuota
                       = $KUOTA_SKS_PER_DOSEN",
"Hari tsb bertepatan dg hari libur kelas tsb", "Pada
   waktu tsb, dosen sedang mengajar di kelas lain",
"Pada jam tsb, kelas ini sedang ada kuliah", "Pada
                       jam tsb, ruangan ini sedang dipakai.");
```

Listing program untuk melakukan proses penjadwalan secara otomatis terdapat dalam fungsi – fungsi sebagai berikut :

a. Mengecek kesanggupan dosen mengajar pada database yang diinputkan dari form kesanggupan dosen mengajar, fungsi terdapat pada ***"function getHariAvailDosen($id_dosen)"***

b. Mengecek hari libur kelas yang terdapat pada fungsi ***"function getHariLibur($id_kelas)"***

c. Mencari waktu kesanggupan dosen mengajar yang tidak berbenturan dengan hari libur kelas yang bersangkutan, fungsi ini terdapat pada ***"function getHariLibur($id_kelas)"***

d. Memeriksa apakah pada jam mulai sampai dengan jam selesai dosen tersebut telah dialokasikan untuk mengajar, fungsi ini terdapat pada ***"function isWaktuAvailDosen($id_dosen, $id_hari, $jam_mulai_sec, $jam_selesai_sec, $id_jadual=0)"***

e. Mencari waktu mulai dan waktu selesai kuliah pada hari $id_hari selama $jml_sks * 50 menit, fungsi ini terdapat pada ***"function getWaktuAvail($id_dosen, $id_hari, $id_mtk, $is_kelas_pagi)"***

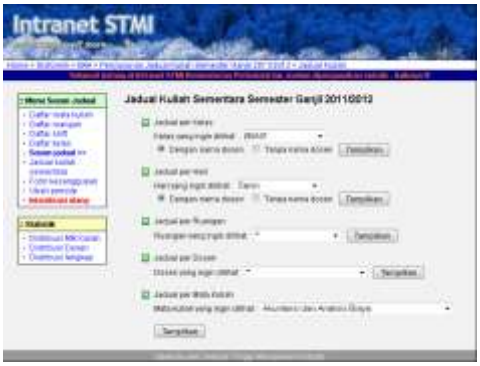
f. Memeriksa apakah ruangan sedang tidak dipakai, fungsi ini terdapat pada ***"function isRuanganAvail($id_ruangan, $id_hari, $jam_mulai, $jam_selesai, $id_jadual)"***

g. Mencari ruangan yang tidak dipakai, fungsi ini terdapat pada ***"function getRuanganAvail($id_hari, $jam_mulai, $jam_selesai, $urut_dari_atas=0, $id_ruangan_preferred=0)"***

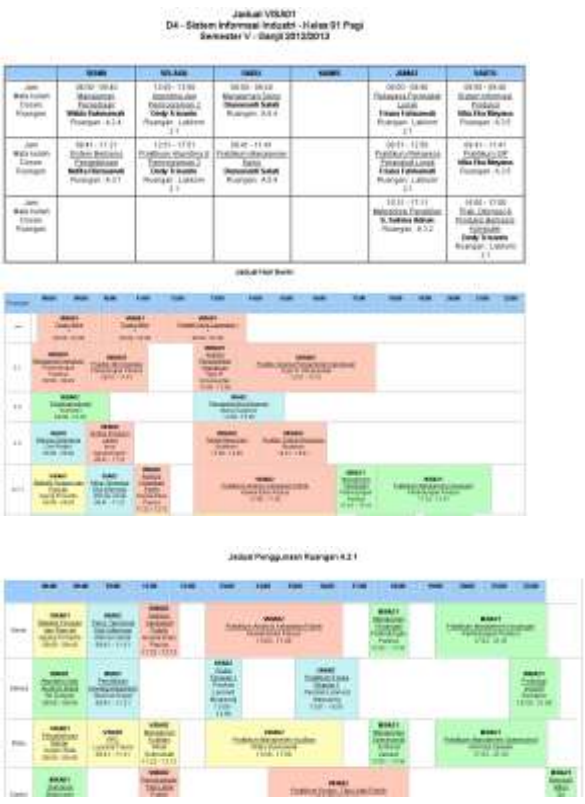
h. Memeriksa dan menjumlahkan SKS yang telah didapat oleh dosen pengampu, fungsi ini terdapat pada ***"function getJmlSKS($id_dosen)"***

i. Memeriksa apakah terdapat kelas atau perkuliahan pada jam tersebut, fungsi ini terdapat pada ***"function isKelasAvail($id_kelas, $id_hari, $jam_mulai, $jam_selesai, $id_jadual=0)"***

Setelah setiap tahapan dan batasan-batasan program berhasil dilalui maka selanjutnya adalah proses pengecekan dan pencetakan dengan tampilan sebagai berikut:



Kemudian untuk menampilkan jadwal kuliah dengan sebelumnya memilih terlebih dahulu menu tampilan penjadwalan kuliah berdasarkan menu dialog diatas, dengan tampilan adalah sebagai berikut:



5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Dengan menggunakan teknik heuristic search yang dikombinasikan dengan teknik smart back tracking dan look ahead penyusunan penjadwalan mata kuliah dapat dioptimalkan. Program dapat mencari solusi penjadwalan pada waktu yang dapat digunakan baik oleh dosen, kelas maupun ruangan yang terlibat dalam suatu mata kuliah. Di samping itu, program dapat meminimalkan tingginya frekuensi mengajar seorang dosen, frekuensi kuliah suatu kelas dan faktor-faktor pengaruh lainnya.

2. Proses penjadwalan mata kuliah menggunakan menggunakan teknik heuristic search yang dikombinasikan dengan teknik smart back tracking dan look ahead ini dapat diterapkan pada kasus-kasus penjadwalan dengan multi angkatan dan multi ruangan.

3. Dengan menggunakan metode *best fitness*, maka teknik heuristic search yang dikombinasikan dengan teknik smart back tracking dan look ahead akan selalu menunjukkan kenaikan *fitness* atau dengan kata lain penjadwalan selanjutnya lebih baik atau minimal sama dengan penjadwalan sebelumnya

5.2 Saran

1. Perubahan nilai bobot dan jumlah mata kuliah saat mutasi tidak akan membawa pengaruh pada kecepatan teknik heuristic search yang dikombinasikan dengan teknik smart back tracking dan look ahead dalam melakukan pencarian solusi optimal, tetapi berpengaruh pada hasil akhir yang dicapai pada akhir penjadwalan. Dapat dilakukan suatu penelitian nilai bobot dan jumlah mata kuliah saat mutasi yang dapat memaksimalkan hasil akhir dari proses penjadwalan menggunakan teknik heuristic search yang dikombinasikan dengan teknik smart back tracking dan look ahead ini.
2. Program penjadwalan mata kuliah ini dapat disempurnakan agar dapat memberikan output akhir tidak hanya berupa jadwal kuliah saja tetapi juga termasuk berita acara

   perkuliahan, jadwal pemakaian ruang dan arsip-arsip serupa lainnya

PUSTAKA

[1] Koza, J., 2001, **Genetic Algorithm**, http://cs.felk.cvut.cz/~xobitko/ga/ intro.html

[2] Munir, Rinaldi, 2006, **Strategi Algoritmik**, Informatika ITB, Bandung

[3] Roman Bartak, 2005, **Foundations of Constraint Satisfaction**, http://kti.mff.cuni.cz/~bartak/, Charles University in Prague

[4] Wahyono, T., 2004, **Sistem Informasi (Konsep Dasar, Analisis Desain dan Implementasi)**, Graha Ilmu, Yogyakarta

[5] www.robertsetiadi.net/articles/snkk.htm, Indonesia